プログラミング言語
第一回

担当：篠沢 佳久
櫻井 彰人

平成21年度：春学期

クラス分け

クラス分け①

- プログラミング言語は二つの教室で同時に行います
  - 703教室（50人収容）
  - 704教室（100人収容）
- どちらの教室も同じ内容の講義をします

クラス分け②

- 703
  - までの学籍番号の学生
- 704
  - 以降の学籍番号の学生
  - 管理工学科2年生以外の学生

講義のガイダンス

講義の目的，進め方

この講義の目指すもの　Part1

- プログラミングの基礎を理解
  - プログラミングの基礎知識を中心に
    - プログラムとは
    - プログラムの実行とは
    - 命令とデータ
    - 判断と分岐
    - プログラミングの構造と実行制御

この講義の目指すもの　Part2

- プログラミングという行為
  - 書く、テストする、使う
- プログラミングがひとりでできるように
  - アルゴリズム
  - データ構造
- プログラミング言語とは
- プログラミングの基本をプログラム言語Rubyを通して学ぶ

この講義の目指すもの　Part3

- Ruby言語でプログラムが作れるように
  - 基本的な演算
  - 選択
  - 繰り返し
  - 配列
  - 関数
  - ファイル入出力・文字列操作

Rubyとは何か?

- Ruby:まつもとゆきひろ氏による、便利さと容易さを兼ね備えたオブジェクト指向スクリプト言語
  - まつもとゆきひろ: http://www.rubyist.net/~matz/
  - スクリプト言語: 動作内容を、台本（Script）のように記述するための、簡易的なプログラミング言語の総称
  - かなり簡単に（周辺環境が）インストールできる
    - 皆さんのコンピュータで実習ができる
  - かなり簡単にプログラムできる
    - 初心者にも容易に学習できる
    - 結構まともに動くプログラムも書ける
  - Ruby on Rails により、Webアプリが容易に書ける
    - そして、Ruby が有名になった

この講義では

- 演習をできる限り行います。
  - そのためには、実は、Ruby プログラムを実行するシステムとして irb をよく用います。
  - irb （interactive Ruby）は対話的に Ruby プログラムを実行するもので、ちょっと実習をするには、適しているのです。

この講義で目指せたら

- もう少し先に行くと
  - ファイル処理
  - DB処理
  - Web アプリケーション
  - 日本語処理

内容に関する注意

- 基本的（初歩的）なことに注力する
- ただし、ところどころ細かい話もする
  - 少し深いことを知りたい方への追加
    - 疑問に対する答えとして
  - 初級者は無視をしてよい

進め方:

- (繰り返しになりますが)Ruby を使う
  - 特に irb を用いて実習を多く
- ある事例(課題)を考える
  - ある動作をする「プログラム」
  - もちろん、簡単版

管理工学科におけるプログラミングの講義

- 本講義(2年春)
  - Rubyを対象として，プログラミングの基礎を中心に
- Java言語，オブジェクト指向
  - ソフトウェア工学(2年秋，飯島先生)
  - ソフトウェア工学実習(3年春，飯島先生)
- プログラミングの応用
  - 管理工学実験演習Ⅱ 計算機(COM)実験(3年通年)

実習について

- この講義では理解を深めるために実習を交えて行ないます。
- 教室…日吉ITC 地下一階
  - 703(50人収容)
  - 704(100人収容)
  - どちらも同じ講義内容

成績について

- 成績のつけかた
  - 講義以外の時間にレポートを作成
    - 5回程度を予定
    - 最後の一回は講義中に実施
  - 講義中の演習問題(平常点)
  - 平常点＋レポートの成績から判定
  - Rubyでプログラムが書ける(自信のある)人は、授業に出席しなくてもレポートさえ出せば単位がとれる
    - 予め申告することが条件
    - 最後のレポートは必ず受けに来て下さい

講義に関する情報

- 講義資料のURL
  - http://www.sakurai.comp.ae.keio.ac.jp/class.html
- 教員，TAへの質問
  - 電子メール
  - 直接質問(アポイント必要)

プログラムとは

プログラミングの必要性
プログラムとプログラム言語

なぜプログラミング？

- 他の講義・実験・演習、卒論に必要
- 必要な技術
- 知っておくべき技術
- 論理的思考力の訓練

プログラムとは①

- 日常使う「プログラム」はどのような意味か？
- すなわち、手順・動作を記した書類
  - 書類といっても、紙に書かれているわけではない

プログラムとは②

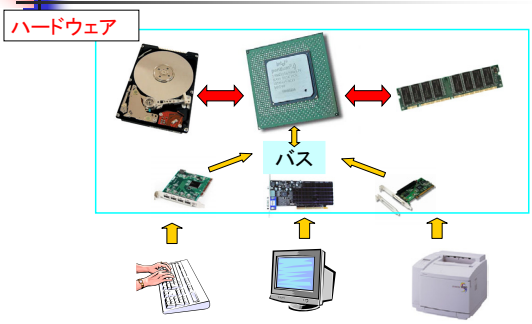
- コンピュータにおいて用いる「プログラム」とは？
- コンピュータが行う動作を
  - 事細かに
  - 逐一

  記述したもの

プログラムとは③

- コンピュータの「記憶装置」に蓄えられている
  - メモリ: 普通はコンピュータの中に隠されている。
  - 内容を持ち運びたいときに、USBメモリとかCD-Rとか DVD-RAMとかいったものにコピーする
- すなわち、プログラムは「ソフトウェア（軟件）」
  - ハードウェア（硬件）ではない
  - つまり、触って感じる物ではない

コンピュータとは



プログラム プログラム プログラム

- OS: プログラムの実行の制御や、ハードウェアの制御と管理など、コンピュータを安全にそして効率良く働かせるための基本ソフトウェア。
  - 例：Windows 2000/XP/Vista, UNIX, Linux, FreeBSD, Solaris, Tron,
- アプリケーション
  - 例: 表計算、文書作成、Java
- ユーザ作成プログラム
  - 例: 「こんにちは」プログラム

プログラムは何語で書くか

- 「書類」だから、記述する言語が必要
  - 言語: 意味のある文字列
- 日本語や英語がだめなことは、勿論
  - なぜか？
- コンピュータが分かる言語？
  - 比喩が過ぎる。コンピュータは意味は分からないから。
  - コンピュータが、文字列から自分がすべき動作に変換できればよい
- コンピュータ用の言語を作ればよい

それをプログラム言語という

- コンピュータは、メモリのどこかに書いてある「命令」を自分の動作に変換すればよい。
  - この「命令」の構成規則が言語
  - この変換規則は言語ではない
    - コンピュータ(機械)にとっては言語(かな？)なので、機械語といったりする
  - この変換規則の例:
    - 01100 → 出力電圧を5Vに
      某神経細胞on → 右手親指曲る (人間の脳)

プログラミング言語とは

- 人間の思いをコンピュータに伝える言葉
  - といったって相手はコンピュータですから
  - 人間の言葉より、機械の言葉にずっと近い。ということは
    - 硬い。すなわち、規則にやかましい
    - 手書き文字ではない。すなわち、キーボード入力

どんなものがあるか？

- 高級(high-level)言語
  - 実行方法による分類
    - コンパイラ言語
      - Ex. Java
    - インタプリタ言語
      - Ex. Ruby
  - 概念による分類
    - 命令型言語
      - Ex. Ruby, Java
    - 関数型言語
  - 分類にならない分類
    - オブジェクト指向言語
      - Ex. Ruby, Java
- アセンブリ言語・機械語

Ruby の長所・短所

- 長所
  - 始めやすい
  - インストールが簡単
  - プログラムもその実行も簡単
  - 一行から始められる
  - (実は隠れた長所がたくさんあります。急成長中)
- 短所
  - 「作法」「行儀」が学びにくい
  - 個性が非常に強い
  - 高速な実行に向かない
  - 大きなプログラムが作りにくい

プログラミング実習

Rubyプログラムの作成手順

“こんにちは”のRubyプログラム

これより右側の文字は、Ruby は無視します。

```
#ファイル名 sample1.rb とします
puts "こんにちは"
```

#から行末までをコメントと見なします。

日本語以外は半角文字で書いて下さい
全角の空白は使わないで下さい
” ” は半角文字で書いて下さい

まずは、やってみよう

- 皆さんの「マイドキュメント」は、日吉のPCでは、Hドライブになっています。
- そこに、Ruby という名のフォルダを作ってください（半角文字として下さい）。

ディレクトリ/フォルダとは

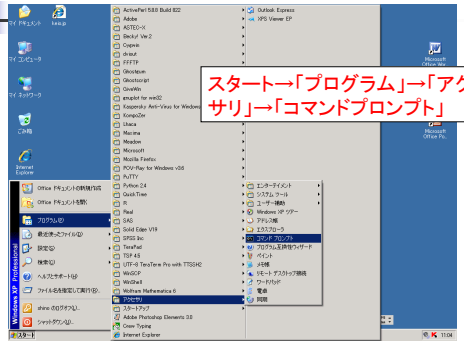
- ハードディスクやCD-ROMなどの記憶装置において、ファイルを分類・整理するための保管場所。
- UNIXやMS-DOSではディレクトリといい、MacintoshやWindowsではフォルダいう。
- Windows の GUI では  のように見えるもの

どうすれば、プログラムを書いたことになるの？

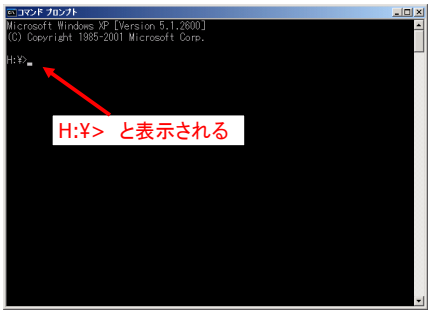
- Ruby 言語の場合

①「**コマンドプロンプト**」というプログラムを起動して行う
② メモ帳（でなくてもいいが）で、プログラムを書く（キーボードから入力する）
③ ファイルにセーブ（ハードディスクに入れること）
  - 仮に sample1.rb（全て小文字）という名前だとしよう
④「コマンドプロンプト」上でruby sample1.rb と入力。
⑤ エラーがなければ結果が得られる

① コマンドプロンプトの起動



コマンドプロンプトの画面



プログラムの書き方

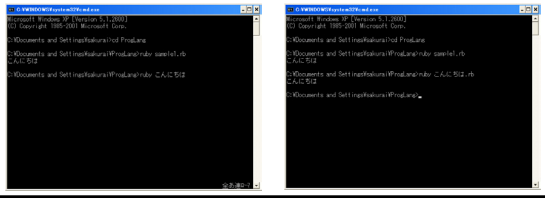
- 二つの作成手順を紹介します
- 初心者はファイルの「拡張子」で混乱します
- どちらの方法でもよいので慣れて下さい

ファイル

- ハードディスクやCD-ROMなどの記憶装置に記録されたデータのまとまり。
- OS（Windows OSなど）は記憶装置上のデータをファイル単位で管理する
- プログラムはファイルに記述する

ファイル名

- 識別のために、ファイルにつけられた名前。一つのディレクトリでは、一名一ファイル
- Windows は、大文字・小文字を区別しない。
- 日本語Windowsでは、かな・カナ・漢字も使える。
  - 入力は、Alt + 半角/全角。

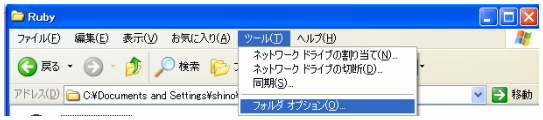


拡張子

- ファイル名の末尾にファイルの種類をあらわす「拡張子」と呼ばれる数文字のアルファベットを付加するのが普通
- ただし、Windows が拡張子を（真剣に！）見るのは、ファイル・アイコンがダブルクリックされたとき
  - ダブルクリックしたときに、メモ帳を起動したいなら ff.txt と、MsWord を起動したいなら ff.doc とする
- Windows ではインストール直後は、拡張子があっても拡張子を表示しなくなっている。便利なようで、極めて不便。
- 皆さんは、当然、表示する方を選ぶ

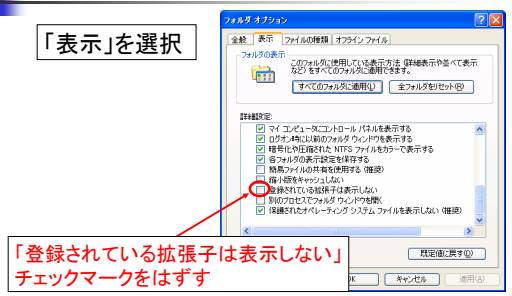
拡張子を常に表示する方法①

フォルダのメニューバー→「ツール」
→「フォルダーオプション」を選択



拡張子を常に表示する方法②

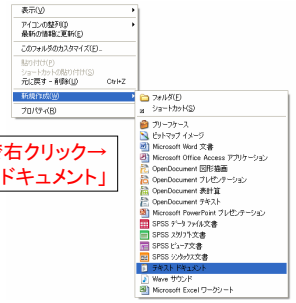
「表示」を選択



「登録されている拡張子は表示しない」
チェックマークをはずす

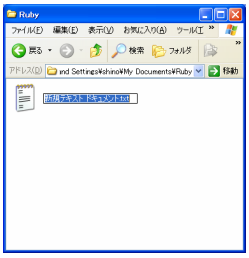
プログラムの書き方その①

プログラムの記述方法①



「Ruby」のフォルダー内で右クリック→「新規作成」→「テキストドキュメント」

プログラムの記述方法②



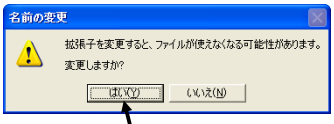
ファイル名の変更
「新規テキスト ドキュメント.txt」
から
「sample1.rb」
に変更する

ファイル名の変更方法①

- ファイルアイコンの下にあるファイル名をゆっくり二回左クリックする
- 右クリック →「名前の変更(M)」
- ファイルの名前を sample1.rb としてください。
  - 半角文字
  - 今回の講義では、拡張子(この例でいえば(.rb)は .rb でなくても(.txt でも)問題はおこらない(はず)。

ファイル名の変更方法②

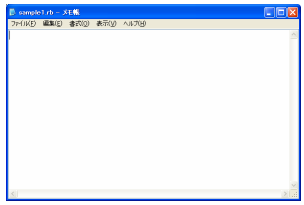
ファイル名を変更すると



「はい(Y)」をクリック→ファイル名が変更される

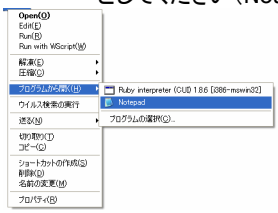
エディターの起動①

- sample1.rbが  というアイコンであれば、ダブルクリックしてください。
  - メモ帳が立ち上がるでしょう、きっと。

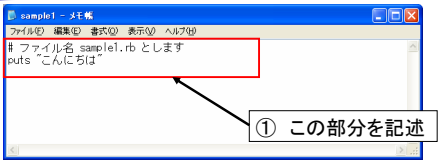


エディターの起動②

- 別のアイコンのときには、
  - 右クリック → プログラムから開く → Notepad としてください（Notepad = メモ帳）



プログラムの書き込み



書き終わったら，上書き保存を行なう

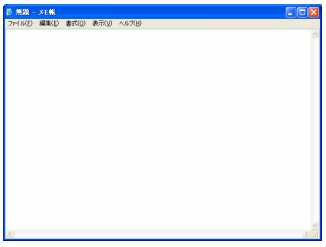
② メニューバーの「ファイル」→「上書き保存」

作成したファイルがRubyプログラム

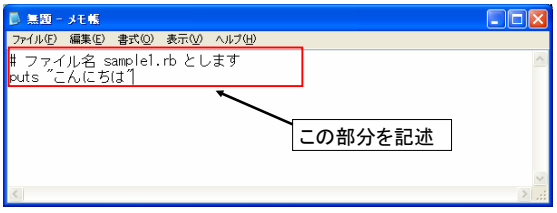
プログラムの書き方その②

エディターの起動

- 「スタート」→「プログラム」→「アクセサリ」→「メモ帳」

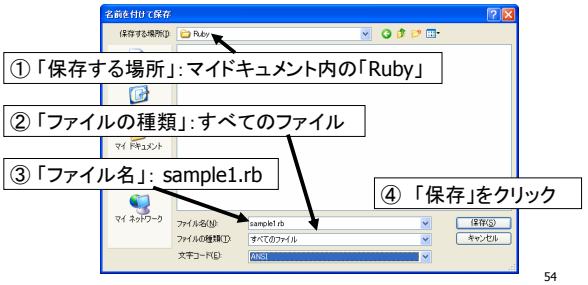


プログラムの記述



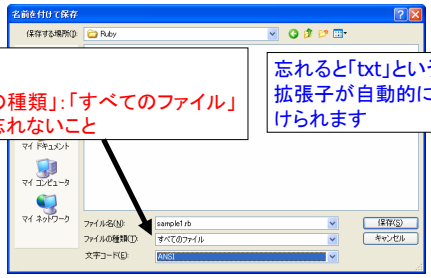
プログラムの保存①

- メニューバーの「ファイル」→「名前を付けて保存」



プログラムの保存②

- メニューバーの「ファイル」→「名前を付けて保存」

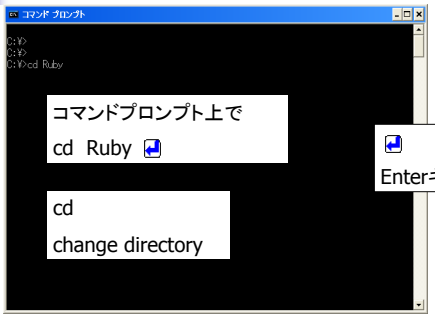


重要!

「ファイルの種類」:「すべてのファイル」の選択を忘れないこと

忘れると「txt」という拡張子が自動的に付けられます

Rubyプログラムの実行①



コマンドプロンプト上で

cd Ruby ↵

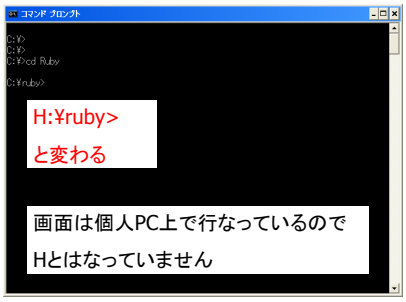
↵

Enterキー

cd

change directory

Rubyプログラムの実行②



H:¥ruby>

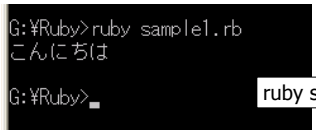
と変わる

画面は個人PC上で行なっているので

Hとはなっていません

Rubyプログラムの実行②

- ruby とは Ruby プログラムを実行するコマンド
  - 指定されたファイルの中身をみて、それに従った動作をする



ruby sample1.rb ↵

Ruby プログラムの実行

ruby Rubyプログラム

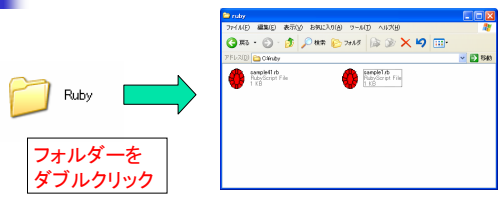
GUI (graphical user interface)

- 表示として、グラフィックスを用いたユーザインタフェイス。入力は、マウスやそれと類似した装置を用いる。
- パソコンでは、Macintosh が使い始めた。
- 今では、これが常識

CLI or CUI (command line user interface)

- 表示として、文字列を用いたユーザインタフェイス
- 入力はキーボードを用いる。
- 入力するものは、コンピュータに対するコマンドであり、行(ライン)単位に入力する。入力する場所をコマンドラインという。コマンドの実行結果はコマンド入力直後に表示する。画面を使い切ると、スクロールする。
- Windows Vista/XP/2000 では、コマンドプロンプトという言葉が用いられる
  - コマンドプロンプトは、本来は、コマンドラインの先頭にコンピュータが書く文字である

コマンドプロンプト①



コマンドプロンプト上で
H:> cd Ruby ↵

コマンドプロンプト②

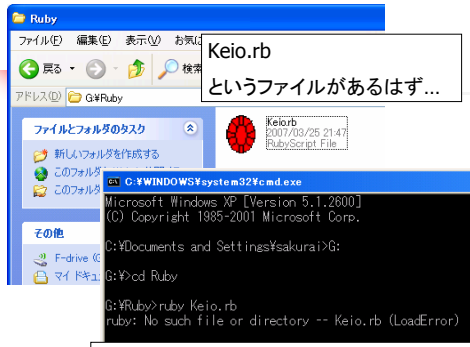
dir ↵ と入力

```
C:¥Ruby>dir
ドライブ C のボリューム ラベルがありません。
ボリューム シリアル番号は 74B7-7992 です

 C:¥ruby のディレクトリ

2008/04/03  17:47    <DIR>          .
2008/04/03  17:47    <DIR>          ..
2008/04/03  18:15                51 sample1.rb
2008/03/03  16:14               103 sample41.rb
               2 個のファイル             154 バイト
               2 個のディレクトリ  229,682,393,088 バイトの空き領域
```

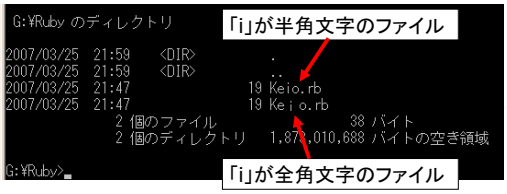
sample1.rbは2008/4/3 18:15に作
51バイトのファイル

文字について

- 日本語 Windows が取り扱う文字には、1バイトコード（所謂半角文字）と2バイトコード（所謂全角文字）とがある。
- 昔は、本当に、半角と全角で表示されていたので分かりやすかったが、今では、プロポーショナルフォントなどを用いるので、分かりにくい。
  - 例：ＡＡと並べれば分かるが KEIO keio
- コンピュータはちゃんと区別するからやっかいだ



エラーメッセージ
「Keio.rb」というファイルがないというエラーが表示



フォルダー内では見分けが付かない

空白について

- Ruby にとって、空白は区切り文字。連続する空白は一つの空白と同じ。しかし、
- Ruby が空白とみなす空白は1バイトコードの空白だけ。2バイトコードの空白は Ruby にとっては空白ではない。
  - よく読んでください。決して、禅問答ではありません。
- どっちの空白かは、人間がみて区別しにくいので、ちょっと目には訳の分からないこと、しかし、よく考えれば分かることが起こる

```
# Keio1.rb
puts    "こんにちは"
```

```
G:¥Ruby>ruby Keio1.rb
こんにちは

G:¥Ruby>ruby Keio2.rb
Keio2.rb:1: Invalid char `¥201' in expression
Keio2.rb:1: syntax error, unexpected $undefined, expecting $end
puts    "こんにちは"
        ^

G:¥Ruby>
```

```
# Keio2.rb
puts  "こんにちは"
```

全角の空白が含まれている

どこが間違っているでしょうか①

```
# Keio3.rb
puts "こんにちは"
```

```
C:¥Ruby>ruby Keio3.rb
Keio3.rb:2: Invalid char `¥202' in expression
Keio3.rb:2: Invalid char `¥223' in expression
```


どこが間違っているでしょうか②

```
C:¥Ruby>ruby Keio3.rb
Keio3.rb:2: Invalid char `¥202' in expression
Keio3.rb:2: Invalid char `¥223' in expression
```

```
# Keio3.rb
puts"こんにちは"
```

半角ではなく全角のsとなっている

どこが間違っているでしょうか③

```
# Keio4.rb
puts    "こんにちは”
```

```
G:¥Ruby>ruby Keio4.rb
Keio4.rb:2: unterminated string meets end of file

G:¥Ruby>
```


どこが間違っているでしょうか④

```
G:¥Ruby>ruby Keio4.rb
Keio4.rb:2: unterminated string meets end of file

G:¥Ruby>
```

```
# Keio4.rb
puts    "こんにちは”
```

全角文字


```
# Keio5.rb
puts   "こんにちは"
```

```
G:¥Ruby>ruby Keio5.rb
Keio5.rb:2: Invalid char `¥201' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥261' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥361' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥311' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥277' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥315' in expression
Keio5.rb:2: unterminated string meets end of file
Keio5.rb:2: warning: parenthesize argument(s) for future version

G:¥Ruby>
```

```
G:¥Ruby>ruby Keio5.rb
Keio5.rb:2: Invalid char `¥201' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥261' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥361' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥311' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥277' in expression
Keio5.rb:2: Invalid char `¥202' in expression
Keio5.rb:2: Invalid char `¥315' in expression
Keio5.rb:2: unterminated string meets end of file
Keio5.rb:2: warning: parenthesize argument(s) for future version

G:¥Ruby>
```

```
# Keio5.rb
puts  "こんにちは"
```

全角文字


```
C:¥Ruby>ruby Keio6.rb
Keio6.rb:1: Invalid char `¥201' in expression
Keio6.rb:1: Invalid char `¥224' in expression
Keio6.rb:2: Invalid char `¥201' in expression
Keio6.rb:2: Invalid char `¥202' in expression
Keio6.rb:2: Invalid char `¥261' in expression
Keio6.rb:2: Invalid char `¥202' in expression
Keio6.rb:2: Invalid char `¥361' in expression
Keio6.rb:2: Invalid char `¥202' in expression
Keio6.rb:2: Invalid char `¥311' in expression
Keio6.rb:2: Invalid char `¥202' in expression
Keio6.rb:2: Invalid char `¥277' in expression
Keio6.rb:2: Invalid char `¥202' in expression
Keio6.rb:2: Invalid char `¥315' in expression
Keio6.rb:2: unterminated string meets end of file
Keio6.rb:2: warning: parenthesize argument(s) for future version
```

全角文字

```
# Keio6.rb
puts  "こんにちは"
```


コンピュータは忠実である

- 言われたとおりに、実行する
- 規則通りに書かれていない場合は，実行せずに，エラーメッセージを出力する
- 書かれたように読む
  - 決して、「きっとこう書きたかったのだろうなあ」と考えて読むことはしない
  - 勿論、「『きっとこう書きたかったのだろうなあ』」と考えて読む」ようにプログラムを書けば、そう書いた範囲で「考えて読む」ようにはなる

プログラム構文上の大原則

- 括弧（広い意味での括弧です）は、開いたら、必ず閉じる。
  - Ruby での例外：「#」で始まるコメント（プログラムと関係のない書き込み）は、改行（そして改行のみ）が閉じる記号
- 複数種の括弧が混じるときには、互いに交錯してはならない
  - 例：{ ([ ]) }
  - 誤例：{] [( )] }

半角文字と全角文字

- プログラムは半角文字で書く
- # の後はコメントであり，全角文字を使用してもよい
- " " の中は全角文字を使用してもよい

練習問題

他の例題（同じようにプログラミングしてみて下さい）

ファイル名

```
# sample12.rb
print "春の"
print "うららの"
puts "隅田川"
```

```
G:¥Ruby>ruby sample12.rb
春のうららの隅田川

G:¥Ruby>
```

```
# sample13.rb
print "春の" + "うららの"
puts "隅田川"
```

```
G:¥Ruby>ruby sample13.rb
春のうららの隅田川

G:¥Ruby>
```

```
# sample14.rb
puts "春の"
puts "うららの"
puts "隅田川"
```

```
G:¥Ruby>ruby sample14.rb
春の
うららの
隅田川

G:¥Ruby>
```

他の例題

ファイル名

```
# sample15.rb
a=5
b=4
print "a+b=" , a+b , "¥n"
print "a-b=" , a-b , "¥n"
print "a*b=" , a*b , "¥n"
print "a/b=" , a/b , "¥n"
```

```
C:¥ruby>ruby sample15.rb
a+b=9
a-b=1
a*b=20
a/b=1
```


Rubyに関する情報

Ruby 関連サイト

- Official site:
  http://www.ruby-lang.org/ja/
  - 「ドキュメント」→本文中の「チュートリアル」とすると「プログラミング入門 」
  - 「ドキュメント」→本文中の「リファレンスマニュアル」
  - 「ダウンロード」→本文中の「ActiveScriptRuby」または「Ruby-mswin32」
- http://www.namaraii.com/rubytips/ も便利です

参考書

- 各種出ています。ご自分の気に入ったものでよいと思います。on-line文書もあります。
  - プログラミング入門 - Rubyを使って -
    - http://www1.tf.chiba-u.jp/~shin/tutorial/

Rubyのインストール

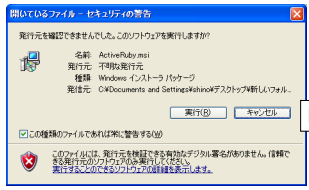
個人PCへのRubyのインストール

Ruby のインストール①

- ActiveScriptRuby
- もっとも簡単。ただし、administrator 権限が必要（自分のPCならOK）
- ActiveScriptRuby をインストールする
  - http://www.ruby-lang.org/ja/
  - 「ダウンロード」→Windows版Rubyバイナリの「ActiveScriptRuby」

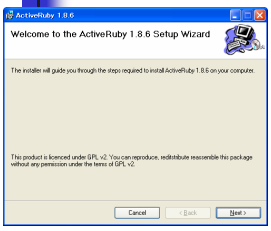
ActiveScriptRubyのインストール①

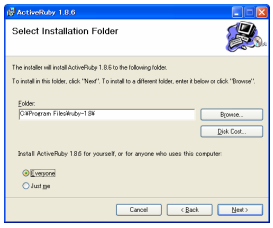
ダウンロードしてきたファイルをダブルクリック



「実行」をクリック

ActiveScriptRubyのインストール②

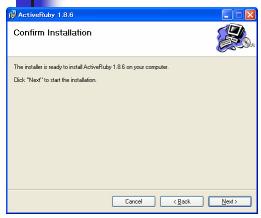


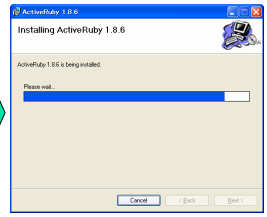


「Next」をクリック

「Next」をクリック

ActiveScriptRubyのインストール③

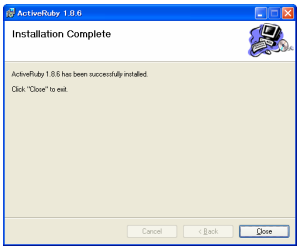




「Next」をクリック

インストール中...

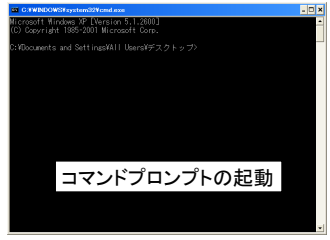
ActiveScriptRubyのインストール④



インストール終了

ActiveScriptRubyのインストール⑤

ダブルクリック



コマンドプロンプトの起動

日吉ITCとはコマンドプロンプトを起動した時のドライブ，フォルダーが異なるの注意して下さい

Ruby のインストール②

- mswin32 を用いる方法（2009年3月現在）
  - ruby-1.9.1-p0-i386-mswin32.zip を
    http://www.garbagecollect.jp/ruby/mswin32/ja/
    よりダウンロード
  - readline.dll を
    http://jarp.does.notwork.org/win32/readline-4.3-2-mswin32.zip
    よりダウンロード

関連情報

http://d.hatena.ne.jp/benikujyaku/20070311
http://asaasa.tk/wiki/index.php?Ruby%2F%E5%89%8D%E6%BA%96%E5%82%99%2FWindows